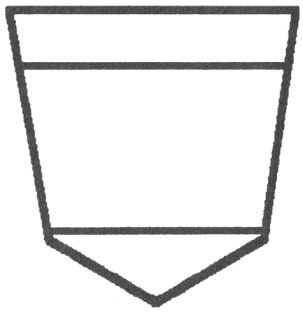
POCKET COMPUTER NEWSLETTER



© Copyright 1985 POCKET COMPUTER NEWSLETTER

Issue 44 - December

FOR PC-1250/51/60/61 USERS

MACHINE CODE TIMING

While *Rick Wenger, 6221 - 18th Avenue, Kenosha, WI 53140,* was working on decoding the machine language instruction set of the SC61860 CPU, he gleaned the following information.

Every opcode tested out (on a Sharp PC-1250) at a multiple of 5.2 microseconds (1 microsecond = 1 millionth of a second). Thus, that is the time required for one machine cycle. Note that all instructions require more than one such cycle, as detailed in the listing that follows. (*Editor's note: On a PC-1350, the cycle time appears to be approximately 4 microseconds.*)

This timing information can be of value to machine language programmers who want to develop routines that operate in a precise amount of time or who want to work towards achieving minimum program execution times. The number of cycles required by an instruction is given. Multiply this by 5.2 microseconds to find the time required to execute an instruction in a PC-1250/1251.

Load Instructions:

LDA #nn	4
LDA P	2
LDA P'	2
LDA SP	2
LDA (P)	2
LDA (W)	3
LDB #nn	4
LDC0 #nn	4
LDC1 #nn	4
LDP #nn	4
LDP A	2
LDP' nn	4
LDP' A	2
LDSP A	2
LDW #nnnn	8
LDWL #nn	5
LD(P) (W)	3
LD(W) A	2
LD(W) (P)	2

Iterative Load Instructions:

LDDA (Y)	7
LDD(X) A	6
LDIA (Y)	7
LDI(X) A	6
LDO(P) (W)	5+4*C0
LDO(P) A	5+1*C0
LDO(P) HR	11+4*C0
LDO(W) A	4+3*C0
LD1(P) (P')	5+2*C1
LD1(P) (W)	5+4*C1

Short Form Loads:
All SLP instructions require 2 cycles.

Branch Instructions:
All absolute branches require 6 cycles.
All forward and reverse relative branches require 7 cycles if the branch actually takes place. Only 4 cycles are used if the test condition is not met.
The RDR instruction takes 10 cycles if the branch is taken, 7 cycles otherwise.

Exchange Instructions:

EXA B	3
EXA (P)	3
EXO(P) (P')	6+3*C0
EXO(P) (W)	7+6*C0
EX1(P) (P')	6+3*C1
EX1(P) (W)	7+6*C1

Increments And Decrements:

INCA/DEA	4
INCB/DEB	4
INCC0/DEC0	4
INCC1/DEC1	4
INCP/DEP	2
INCX/DEX	6
INCY/DEY	6
INC/DE(08)	4
INC/DE(09)	4
INC/DE(0A)	4
INC/DE(0B)	4

Addition Instructions:

ADDA #nn	4
ADD(P) A	3
ADD(P+) AB	5
ADC(P) A	3
ADD(P) #nn	4
ADEC0(P) A	7+3*C0
ADEC0(P) (P')	7+3*C0

Subtraction Instructions:
Same as corresponding addition instructions.

Rotate Instructions:

RDA	2
RLA	2
RRA	2
RLO (P)	5+1*C0

Another *Personal Information*™ product.

RRO (P)	5+1*CO

Boolean And Compare Operations:

ANDA nn	4
ORA nn	4
BITA nn	4
CPA nn	4
AND(P) A	3
OR(P) A	3
BIT(P) A	3
CP(P) A	3
AND(P) nn	4
OR(P) nn	4
BIT(P) nn	4
CP(P) nn	4
AND(W) nn	6
OR(W) nn	6
BIT(W) nn	6

Subroutine Instructions:

JSR nnnn	8
RTS	4
SS nn	7

I/O Instructions:

OUTC	3
OUTD	2
OUTE	3
OUTF	2
REFR nn	6+nn
In nn	4
INA PORT	2
INA KBD	2
OSIGNAL	Until done
ONO Signal	Until done

Miscellaneous Directives:

SET C	2
CLR C	2
PUSH A	3
POP A	2
PUSH-cc-nnnn	9
LNOP	3
SNOP	2
JCA	7 cycles for each condtion checked an not met. 9 when when a condition is met and a branch ensues. 7 for the final branch if none of the conditions are met.

Example JCA Timing:

PUSH-02-CONT	Number of cycles used by JCA:
JCA 01 XXXX	9 if A = 1
02 XXXX	7 + 9 = 16 if A = 2
XXXX	7 + 7 + 6 = 21 if A<>1 and A<>2

CORRECTION AND CLARIFICATION

Issue 42 of ***PCN*** carried an article that discussed display I/O operations. The top right-hand column on page 4 provided a short listing of a routine that the text said would cause the display to wink. The text also indicated that substituting SNOPs for the first IN 08 directive in that routine would make the winking disappear. These comments concerning the routine are in error.

Indeed, putting SNOPs in place of the first IN 08 instruction simply causes the routine to return immediately to BASIC.

The IN 01 instruction further on in the routine is what *prevents* the display from winking. If you replace that IN 01 directive with two SNOP instructions, then you can observe a pronounced winking of the display.

FOR HEWLETT-PACKARD HP-71B USERS

FINDING FACTORS

Engineers, scientists and mathematicians frequently find it necessary to factor equations. The old trial-and-error method just doesn't compare to having a computer find those prime factors! Here is a program tailored for the HP-71B that can find prime factors, frequently in the blink of an eye. It is an adaptation of a program originally contributed by mathematician *Norlin Rober*.

To put it to use, just load the program and start it up. Respond to the prompt for an integer number to be evaluated.

Program *Finding Factors.*

```
5 DESTROY ALL @ WIDTH 96 @ DELAY 8,.
5 @ A$=KEY$ @ FIX 0
10 INPUT "INTEGER? ";I @ J=I @ M=0 @
 T=2 @ GOSUB 50 @ T=3 @ GOSUB 50 @ T
=5 @ GOSUB 50 @ T=1
20 T=T+6 @ GOSUB 50 @ T=T+4 @ GOSUB
50 @ T=T+2 @ GOSUB 50 @ T=T+4 @ GOSU
B 50 @ T=T+2 @ GOSUB 50
30 T=T+4 @ GOSUB 50 @ T=T+6 @ GOSUB
50 @ T=T+2 @ GOSUB 50 @ IF T*T<=J TH
EN 20
40 GOTO 100
50 IF J/T-INT(J/T) THEN RETURN
60 IF T=I THEN 130
70 M=M+1 @ J=J/T @ IF J/T=INT(J/T) T
HEN 70
80 BEEP 1500,.15 @ U$=STR$(T) @ U$=U
$[1,LEN(U$)-1] @ DISP U$;" IS A FACT
OR..."
85 V$=STR$(M) @ V$=V$[1,LEN(V$)-1] @
 DISP "OCCURRING ";V$;" TIMES."
90 M=0 @ IF T*T<=J THEN RETURN
100 IF I=J THEN 130
110 IF J=1 THEN DISP "END OF LIST" @
 GOTO 5
120 T=J @ J=1 @ M=1 @ GOTO 80
130 BEEP @ W$=STR$(I) @ W$=W$[1,LEN(
W$)-1] @ DISP W$;" IS A PRIME" @ GOT
O 5
```

The HP-71B will then produce the prime factors along with the number of times each factor would be repeated. The program will handle any integer having up to 10 digits.

Here is an example of its operation so that you can verify proper loading of the program (which has a byte count of 586). Computer responses are in bold type:

INTEGER? 13510750
2 IS A FACTOR...
OCCURRING 1 TIMES.
5 IS A FACTOR...
OCCURRING 3 TIMES.
11 IS A FACTOR...
OCCURRING 1 TIMES.
17 IS A FACTOR...
OCCURRING 3 TIMES.
END OF LIST

The program is pretty fast when extracting relatively low value primes. However, if you test it with the number 9999999967 (the largest prime that the program is designed to handle) you can plan on waiting about 20 minutes. A beep sounds each time a factor is extracted so that you do not have to keep an eye on the display when dealing with large values.

CORRECTIONS TO TRIANGLE PROGRAM

Issue 42 of *PCW* presented a program for the HP-71B that solved triangulation problems. Unfortunately, the program contained several errors that may cause problems when attempting to solve certain cases. To correct the program edit line 410 so that it appears as follows and add line 415 as shown here:

```
410 IF K<>I+3 THEN G=A(K-3) @ H=A(9-I-K) @ A(K)=90
    @ GOTO 420
415 GOTO 430
```

Thanks to *Norman Ishler* for being the first to point out the need for these corrections.

FOR PC-1500 & PC-2 USERS

LEAST-SQUARES POLYNOMIALS

This program will determine the least-squares polynomial, of specified degree, fitting a set of given pairs (x,y).

To illustrate, suppose we wish to fit a third-degree polynomial to the following data:

X	Y
8.0	1.8
8.5	2.1
9.0	1.9
10.0	2.2
10.6	2.8
11.0	3.2
12.0	4.8

Begin program execution with RUN. Enter 7 in response to NO OF PAIRS? Enter the given values of X and Y as called for by prompts. For DEGREE OF POLY? enter 3. To examine all possibilities, respond to further prompts by entering Y for yes.

The resulting output will include the coefficients of the terms of the third-degree least-squares polynomial. For this example, the polynomial (after some rounding of coefficients) is: $Y = -24.65 + 9.60X - 1.17X^2 + .05X^3$.

A graph of this polynomial is plotted, with dashed lines marking off the area containing values of X from 8 to 12, and of Y from 1.8 to 4.8. A small circle locates each of the pairs (x,y) used as input data, providing a quick glimpse of how well the curve fits the given data. The coefficient of correlation is also given.

The display will then show a prompt requesting a value of X. As an example, enter 10 and press ENTER. The display will show 2.30810458. This is the value of Y obtained using the least-squares polynomial equation derived by the program. Key ENTER to continue with additional values.

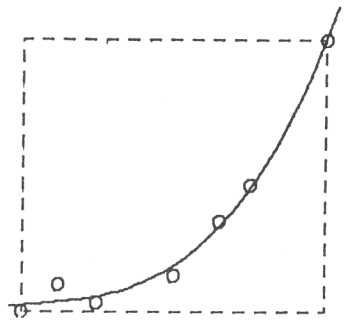
It is possible to find a least-squares polynomial of lower degree, without having to re-enter the initial data. To do this, key DEF N, and enter 2 as the degree. The same options as before will be available for the second-degree polynomial. (Note that it is not possible to specify a degree *higher* than that initially used. Attempting to do so will result in ERROR 9 being displayed.)

Note too, that the polynomial degree may not exceed the number of (x,y) pairs given.

Some Warnings

One should avoid using values of X that are beyond the range

Example *Operation of the Least-Squares Polynomial Program.*



```
INTERVAL FOR X:
         8.00000000
        12.00000000

INTERVAL FOR Y:
         1.80000000
         4.80000000

COEFFICIENTS:
0      -24.64780564
1        9.60166665
2       -1.16610716
3        0.04754995

CORRELATION:
         0.99529243
```

of the X-values included in the given data. Do not assume that a least-squares polynomial will accurately represent the relationship outside the interval of given values. To illustrate, in the example used above the least-squares polynomial should be applied using only values of X that are between 8 and 12.

The temptation to use high-degree polynomials (to obtain a higher correlation) should be resisted, especially when a relatively small number of given (x,y) pairs are used. Although a closer fit to the given data may result, a high-degree polynomial will give overdue attention to small "random" errors in the given data. This is particularly true when the given values of X and Y have been obtained by measurement.

In this program, coefficients are calculated by solving a system of linear equations. This system is often said to be "ill-conditioned," especially when a high degree is specifield for the polynomial. (That is, slight errors in the elements of the matrix, caused by roundoff, will result in relatively large errors in the solution.) One partial cure for this problem is to "code" the input data, by subtracting a constant value from each X.

A slight modification to the program will take care of coding. Change line 22 to:

```
22 X=X(I)-H,T=Y(I)
```

and change line 60 to:

```
60 S=X-H
```

To use the coding, a value must first be assigned to the variable H. This should be approximately the mid-value of the given values of X. (In the example used in this article, a suitable choice for H would be a value of 10.)

It is important to note that the resulting polynomial will give the coefficients of powers of (X-H), rather than of X.

This article was provided by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

Program *Least-Squares Polynomials.*

```
 10 INPUT "NO OF PAIRS? ";M:DIM X(M-1),Y(M-1):GOTO 12
 11 GOTO 10
 12 WAIT 0:FOR I=0TO M-1
 13 CLS :PRINT "X(";STR$ (I+1);")";:INPUT "? ";X(I):GOTO 15
 14 GOTO 13
 15 CLS :PRINT "Y(";STR$ (I+1);")";:INPUT "? ";Y(I):NEXT I:WAIT :CLS :GOTO 17
 16 GOTO 15
 17 INPUT "DEGREE OF POLY? ";N:IF M>NTHEN 20
 18 GOTO 17
 20 CLS :DIM A(N),B(2*N),C(N),M(N,N+1)
 21 FOR I=0TO M-1:A=1
 22 S=X(I),T=Y(I)
 23 FOR J=0TO N:B(J)=B(J)+A,C(J)=C(J)+A*T,A=A*S:NEXT J
 24 FOR J=N+1TO N+N:B(J)=B(J)+A:A=A*S:NEXT J:NEXT I:GOTO 26
 25 "N"INPUT "N? ";N
 26 FOR I=0TO N:FOR J=0TO N:M(I,J)=B(I+J):NEXT J:M(I,N+1)=C(I):NEXT I
 27 FOR I=0TO N-1:FOR J=I+1TO N:A=M(J,I)/M(I,I)
 28 FOR K=I+1TO N+1:M(J,K)=M(J,K)-M(I,K)*A:NEXT K:NEXT J:NEXT I
 29 FOR I=NTO 1STEP -1:A(I)=M(I,N+1)/M(I,I)
 30 FOR J=0TO I-1:M(J,N+1)=M(J,N+1)-M(J,I)*A(I):NEXT J:NEXT I:A(0)=M(0,N+1)/M
 31 A=C(0)/M,D=0,E=0:FOR I=0TO M-1
 32 X=X(I):GOSUB 60:B=Y-A,C=Y(I)-A,D=D+B*B,E=E+C*C:NEXT I:R=SQR (D/E)
 40 USING "######.########"
 42 A$="":INPUT "DISPLAY COEFF (Y/N)? ";A$:IF A$="Y"GOSUB 70
 44 A$="":INPUT "PLOT (Y/N)? ";A$:IF A$="Y"CLS :GOSUB 90
 46 A$="":INPUT "PRINT COEFF (Y/N)? ";A$:IF A$="Y"GOSUB 80
 50 INPUT "X? ";X
 52 GOSUB 60:PRINT Y:GOTO 50
 60 S=X
 62 Y=A(N):FOR J=N-1TO 0STEP -1:Y=S*Y+A(J):NEXT J:RETURN
 70 FOR I=0TO N:PRINT "A(";STR$ I;")=";A(I):NEXT I
 72 PRINT "CORRELATION:",R:RETURN
 80 LPRINT "COEFFICIENTS:"
 82 FOR I=0TO N:LPRINT STR$ I;TAB 3;A(I):NEXT I:LPRINT
 84 LPRINT "CORRELATION:",R:RETURN
 90 A=X(0),B=A,C=Y(0),D=C:FOR I=1TO M-1:IF X(I)<ALET A=X(I):GOTO 94
 92 IF X(I)>BLET B=X(I)
 94 IF Y(I)<CLET C=Y(I):GOTO 98
 96 IF Y(I)>DLET D=Y(I)
 98 NEXT I:E=(B-A)/200,F=180/(D-C)
100 GRAPH :GLCURSOR (8,-240):SORGN :RLINE -(200,180),6,,B:POKE &79EA,0
102 P=0:FOR I=-8TO 208STEP 3:X=A+E*I:GOSUB 60:V=(Y-C)*F
104 IF V>240OR V<-60LET P=0:GOTO 110
106 IF PLINE -(I,V):GOTO 110
108 GLCURSOR (I,V):P=1
110 NEXT I
112 GLCURSOR (0,0):A$="":INPUT "PLOT INPUT DATA (Y/N)? ";A$
114 IF A$<>"Y"THEN 118
116 FOR I=0TO M-1:GLCURSOR ((X(I)-A)/E-4,(Y(I)-C)*F-4):LPRINT "o":NEXT I
118 GLCURSOR (0,-100):TEXT :LPRINT "INTERVAL FOR X:":LPRINT A:LPRINT B
120 LPRINT :LPRINT "INTERVAL FOR Y:":LPRINT C:LPRINT D:LPRINT :RETURN
```

FOR PC-1350 USERS

PC-1350 ROM ROUTINES

OK, all you PC-1350 machine language programmers, here is the dope for which you have been waiting. Provided -- once again -- by that super-PC-sleuth: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* Norlin once again has demonstrated his mastery of the realm-of-ROM by locating those key routines that can be so useful to programmers. The article that follows speaks for itself and Norlin's outstanding ability and dedication to sleuthing-out the internal secrets of pocket computers. Thanks, Norlin, for all you have provided over the years to *PCN* readers!

Getting Started

Here are some routines in the ROM of the Sharp PC-1350 that are useful in machine language programming.

The ROM of the PC-1350 exists in two parts. The first part, which is located from 0000 to 1FFF, is the internal or "hidden" ROM, which is not accessible using PEEK. The second part is located between addresses 8000 to FFFF. It is referred to as the external ROM.

It appears (to date) that any revisions in ROM made by Sharp are in the "external" portion. To use the routines beginning at addresses above 8500, you will need to make sure that you have a computer with the same version of ROM as the one to which these addresses apply. Fortunately, it is easy to check this, since there are some checksum routines built into the PC-1350.

To determine the checksum for the external ROM of your PC-1350, execute CALL 6802A. After four or five seconds, the sum will be displayed. If you get anything other than 39380, your version is different from the one to which the routines in this article apply. Thus, any of the routines described in this article that are located above 8500 may not work on your PC.

Floating-Point Arithmetic

Arithmetic operations involving floating-point numbers are carried out in four floating-point registers. These will be referred to as R1 to R4. These registers occupy a portion of the "Central Processing Array" (CPA). Each register consists of 8 bytes. Register R1 uses CPA registers 10 to 17. R2 uses 18 to 1F, R3 uses 20 to 27, and R4 uses 28 through 2F.

The easiest way to get started with floating-point arithmetic is to obtain input data from a "fixed" variable. That is, from a variable named A through Z. The memory map shown in *PCN* Issue 39 (on page 6) shows the locations of these variables. A calculated result may be transferred back to a fixed variable for printing, display or further use. An example later in this article will illustrate such usage.

Here is a list of the beginning addresses of routines that will manipulate data and perform calculations within the CPA floating-point registers:

```
0142  Clear R1
0136  Clear R2
013C  Clear R3
01B6  Copy R2 into R1
01DE  Copy R3 into R1
01C4  Copy R4 into R1
01D2  Copy R1 into R2
01EF  Copy R1 into R3
01FB  Copy R1 into R4
01E3  Copy R2 into R3
019C  Copy R2 into R4
01A8  Copy R4 into R2
020F  Exchange R1 with R2
```

In the routines involving arithmetic and functions, twelve-digit mantissas are calculated and normalized by each routine. The result, however, is not rounded to 10 digits. The carry flag will be set if an overflow or an illegal input argument is encountered. The trigonometric functions and their inverses are calculated according to the current mode setting (DEGREE, RADIAN or GRAD).

```
8AB8  R2 + R1 into R1   (R3, R4 unaffected)
8ACF  R2 - R1 into R1   (R3, R4 unaffected)
8AD9  R2 * R1 into R1   (R4 unaffected)
8AE3  R2 / R1 into R1   (R4 unaffected)
8AEC  R2 ^ R1 into R1
8084  INT(R1) into R1   (R3, R4 unaffected)
8FF4  ABS(R1) into R1   (R2, R3, R4 unaffected)
8B56  SGN(R1) into R1   (R2, R3, R4 unaffected)
8B09  SQR(R1) into R1   (R4 unaffected)
8AF3  LN(R1)  into R1
8AFB  LOG(R1) into R1
8B02  EXP(R1) into R1
89F1  Change sign of R1
8B10  SIN(R1) into R1
8B17  COS(R1) into R1
8B1E  TAN(R1) into R1
8B25  ASN(R1) into R1
8B2C  ACS(R1) into R1
8B33  ATN(R1) into R1
C554  Set DEGREE Mode
C55C  Set RADIAN Mode
C564  Set GRAD Mode
```

An 8-register floating-point stack is also available, for temporary storage of intermediate results. This stack uses the RAM area 7680-76BF, although the programmer does not need to be concerned with the location. The routines that utilize this stack are as follows:

```
0FE9  Push R1 onto FP stack. Note that the contents
      of R1 will not be retained.
10C1  Pop R1
10AA  Pop R2
```

Certain constants may be put into R1 or R2, in floating-point format, using the following routines:

```
8C41  Load R1 with 1
8B63  Load R1 with -1
0BF0  Load R1 with 12-digit value of Pi
0156  Load R2 with 1
0812  Load R2 with LOG e
081E  Load R2 with 180/Pi
```

There are also routines available for conversion between hexadecimal and decimal. The 2-byte hexadecimal number involved will occupy a portion of the CPA and is designated here as Hexadecimal Register H1. The high-order byte of H1 is located in CPA register 19 and the low-order byte is in CPA register 18. (Note that H1 overlaps a portion of R2. Hence H1 and R2 may not be used simultaneously.)

```
11B0  H1, treated as unsigned binary, converted
      to BCD in R1
11B7  H1, treated as signed binary, converted
      to BCD in R1
163A  R1 converted to unsigned binary, placed
      into H1, carry flag set if out of range
1633  R1 converted to signed binary, placed
      into H1, carry flag set if out of range
```

After a sequence of operations has been performed, the final result should be rounded to 10 digits by calling upon this routine:

```
037A  Round R1 to 10-digit mantissa
```

An Example

This example illustrates how a simple machine language program can be used to perform the equivalent of the BASIC

statement: A = SQR(B*B+C*C).

First it will be necessary to set aside a portion of memory for the machine language program. The beginning of the BASIC area, normally &6030, will be set to &6100. To do this, execute POKE &6F01,0,&61. Then, in the PRO mode, execute the NEW command. The area 6030 through 60FF will then be available as machine language program space.

The accompanying listing shows the instructions needed to perform the calculation mentioned above. The mnemonics are those given in Issue 38 of *PCN*.

The program may be entered into memory using POKEs. You may begin with POKE &6030,0,7,&10,&6C,&F0,&90,&18,&E1,..., and so on.

To use the program, first store the values to be used into the variables B and C. Next, execute CALL &6030. The variable A will then contain the result of the calculation. As a test, execute the program with B=6 and C=7. When you display A you should see 9.219544457.

This example is not intended as a particularly practical use of machine language, but rather as an illustration of how the floating-point routines may be put to use.

The BASIC area may be restored to its normal range by execution of CALL &800F. Note that doing this will also clear all variables and the RESERVE memory.

Program ***Machine Language Calculation of A=SQR(B*B+C*C).***

Address	Codes	Mnemonics	
6030	00 07	LDCO 07	Set counter for 8-byte block
6032	10 6C F0	LDW 6CF0	Point W to fixed variable B
6035	90	SLP 10	Point P to R1 (in CPA)
6036	18	LDO (P)(W)	Copy variable B into R1
6037	E1 D2	SS 01D2	Copy R1 into R2
6039	78 8A D9	JSR 8AD9	Multiply
603C	E1 FB	SS 01FB	Save result in R4
603E	00 07	LDCO 07	Set counter for 8-byte block
6040	10 6C E8	LDW 6CE8	Point W to fixed variable C
6043	90	SLP 10	Point P to R1
6044	18	LDO (P)(W)	Copy variable C into R1
6045	E1 D2	SS 01D2	Copy R1 into R2
6047	78 8A D9	JSR 8AD9	Multiply
604A	E1 A8	SS 01A8	Copy R4 into R2
604C	78 8A B8	JSR 8AB8	Add
604F	78 8B 09	JSR 8B09	Calculate square root
6052	E3 7A	SS 037A	Round result to ten digits
6054	10 6C F8	LDW 6CF8	Point W to fixed variable A
6057	90	SLP 10	Point P to R1
6058	00 07	LDCO 07	Set counter for 8-byte block
605A	19	EXO (P)(W)	Exchange R1 with variable A
605B	37	RTS	Return from ML program

More ROM Info

Some other useful operations that can be performed using ROM routines are shown in the accompanying table. Hope it helps you to further enjoy the use of your PC-1350!

Table ***More PC-1350 ROM Routines.***

Manipulation of bytes

027D Contents of B and A copied into XH and XL
0282 Contents of B and A copied into YH and YL

115C Contents of Y pushed onto Central Processing Array stack; A and B are lost.
1167 Y is popped from CPA stack; A and B lost.

1175 Block transfer of bytes, in RAM. B = number of bytes moved. Original block starts at address Y+1; bytes are moved into area starting at address X+1. The two areas may overlap only if Y > X.

Calculations (in hexadecimal)

Here U will represent the two-byte hexadecimal number whose high-order byte is in B, low-order byte in A. Also, V represents the two-byte number whose high- and low-order bytes are located, respectively, in CPA addresses ØB and ØA.

1EBE V is replaced by its complement. (This is the equivalent of subtraction from &10000.)
183D The product U * V is calculated and put into V. If an overflow is produced, flag C is set.

Power off

04D8 Computer is turned off. Power comes back on when the ON/BRK key is pressed.

BEEP

C318 The number of BEEPS produced will be whatever number is contained in floating-point register R1.

Display

04AD Display off
04B1 Display on

DC3C Numeric data in R1 is displayed, in default format, provided that Ø1 is stored into address 6F15 immediately preceding execution of this routine.

1E0C Clear Display Output Buffer. (This is the area located 6D00-6D5F.)

1DCF Display contents of Display Output Buffer. The display will show the characters whose ASCII codes are in appropriate locations in the Display Output Buffer, as follows:

Top line of display:	6D00 to 6D17
Second line	6D18 to 6D2F
Third line	6D30 to 6D47
Bottom line	6D48 to 6D5F

Delay

BF3C The length of the delay is the complement of the two-byte number contained in CPA registers 09 and 08, in units of 1/64 second. Use of the BRK key will discontinue the delay.

Keyboard Input

ROM subroutine 120A will wait until a key is pressed. Then A will contain the ASCII code for the key used, including codes that require the SHIFT or SML prefix.

The DEF prefix, however, must be disabled prior to calling subroutine 120A. This is done by storing ØØ into 783C. Immediately following subroutine 120A, 1Ø should be stored into 783C.

The codes obtained when the control keys are used are as follows:

CLS	02	OFF Switch	0A
SHIFT CLS	03	INS	0B
Scroll up	04	DEL	0C
Scroll down	05	ENTER	0D
SHIFT ENTER	06	Cursor right	0E
ON/BRK	07	Cursor left	0F
MODE	08	Shifted cursor left	13
		Shifted cursor right	14

Codes obtained for SHIFTed character keys are as follows:

A	81	H	88	S	F3
B	82	J	8A	V	F6
C	83	K	8B	X	F8
D	84	L	8C	Z	FA
F	86	M	8D	SPC	F1
G	87	N	8E	=	F4

This routine may also be used to accept repeated input of a key when the key is held down. This key-repeat feature is in effect whenever 02 is stored into 6F11 prior to execution of subroutine 120A. (Key repeat is turned off automatically when the key is released.)

CE-126P Printer

DC3C Numeric data in R1 is printed and displayed, in default format, if preceded by storing 01 into 6F15 and 14 into 783C. Follow by storing 10 into 783C.

1C16 Printout pointer (6FAE) is reset to zero. This will assure a fresh start for printing, canceling any leftover data resulting from having interrupted a previous printout with BREAK.

1C59 Clear Printout Buffer. This buffer is located 7480-7497.

1C05 Print contents of Printout Buffer (7480-7497) as one line. Non-character codes will be printed as spaces.

A372 The byte contained in A is transmitted to a buffer located in the CE-126P. The capacity of this buffer is 24 bytes. The contents of this buffer are printed when subroutine A51F is executed.

A51F Print the codes that have been transmitted to the CE-126P by subroutine A372. These codes will be printed right-justified. The entire character set of the CE-126P is usable, including Katakana characters (Codes A1 to DF), exponents (codes 80-8C), and various other symbols.

Index *1985 Articles*.

Article	Author	Issue	Page
Announcement: Access Pocketable Portable	PCN, Staff	41	1
Announcement: Motorola Hand-Held Communicates Via Radio Waves	PCN, Staff	39	1
Announcement: Pocket Computer Sports Touch Pad Screen	PCN, Staff	40	1
Announcement: Sharp PC-2500 Portable Notebook Computer	PCN, Staff	42	1
Commentary: Watch Pocket	PCN, Staff	37	8
Commentary: Watch Pocket	PCN, Staff	39	8
Commentary: Watch Pocket	PCN, Staff	40	8
Corrections: See Issue Number 44	Various	44	2, 3, 8
Program: Appointments/Alarms (HP-71B)	PCN, Staff	39	2
Program: Bowling Score Keeper (PC-1500/PC-2)	Sincick, Robert	42	6
Program: Calendar (HP-71B)	PCN, Staff (conversion)	41	5
Program: Chess Clock (PC-1500/PC-2)	Gibson, John	41	6
Program: CLOAD M? (Verify Machine Language Saves) (PC-1500/PC-2)	Bowman, Eric	40	7
Program: Curve-Fitting (HP-71B)	PCN, Staff (conversion)	40	5
Program: Disassembler (PC-1250/51,PC-3/3A)	Wenger, Rick	41	3
Program: Displaying Graphs (PC-1350)	Rober, Norlin	42	8
Program: Finding Prime Factors (HP-71B)	PCN, Staff (conversion)	44	2
Program: Histogram (HP-71B)	PCN, Staff (conversion)	43	2
Program: Least-Squares Polynomials (PC-1500/PC-2)	Rober, Norlin	44	3
Program: Memory Dump (PC-1250/51/60/61,PC-3/3A)	PCN, Staff (conversion)	37	3
Program: Monitor (PC-1250/51,PC-3/3A)	Wenger, Rick	40	2
Program: Monitor (PC-1350)	Wenger, Rick	43	5
Program: Morse Code (HP-71B)	PCN, Staff (conversion)	38	6
Program: Password Protection (PC1500/PC-2)	Bowman, Eric	38	7
Program: Payroll Deductions (HP-71B)	PCN, Staff (conversion)	37	4
Program: Payroll Deductions (PC-1250/51/60/61,PC-3/3A)	PCN, Staff (conversion)	37	2
Program: PC! Solve (PC-1500/PC-2)	Bowman, Eric	39	7
Program: Root Finder (PC-1500/PC-2,PC-1250-61,PC-3/3A,PC-1350)	Wenger, Rick	43	4
Program: Simultaneous Equations (PC-1250-61,PC-3/3A)	PCN, Staff (conversion)	40	4
Program: Triangles (HP-71B)	PCN, Staff (conversion)	42	5
Review: Sharp PC-1350	PCN, Staff	37	1
Review: Tramsoft Toolkit	PCN, Staff	38	1
Technical Info: Converting PC-1500 Programs to PC-1250/PC-3	PCN, Staff	40	4
Technical Info: Inside the PC-1350	Rober, Norlin	41	8
Technical Info: Keyboard & Display I/O (PC-1250/PC-3)	Wenger, Rick	42	2
Technical Info: Machine Code Exposed (PC-1250-61,PC-3/3A,1350)	Wenger, Rick	38	2
Technical Info: Machine Code Timing (PC-1250-61,PC-3/3A,PC-1350)	Wenger, Rick	44	1
Technical Info: Manual Addendun (PC-1250/51,PC-3/3A)	Wenger, Rick	43	1
Technical Info: Memory Map (PC-1261)	Rober, Norlin	39	4
Technical Info: Memory Map (PC-1350)	Rober, Norlin	39	5
Technical Info: Miscellaneous Tips (PC-1350)	Rober, Norlin	40	7
Technical Info: More M.L. Instructions (PC-1250-61,PC-3/3A,1350)	Wenger, Rick	41	2
Technical Info: PC-1350 Welcome Aboard	PCN, Staff	37	7
Technical Info: PC-1500/PC-2 Potpourri	PCN, Staff	37	5
Technical Info: Pocket Computer Speed Comparisons	Rober, Norlin	40	6
Technical Info: ROM Routines (PC-1350)	Rober, Norlin	44	5

A FEW MORE CORRECTIONS

The PC-1250 disassembler listing on page 4, Issue 41, of *PCN* is missing a line of code. Add this line to the listing:

C455: 14 1A 19 8B

Also, be advised that the byte of code at address C44A contains 0A.

PC-1350 Monitor Corrections

There are several items in the PC-1350 Monitor article that was presented in Issue 43 of *PCN* that need changing:

First, on page 6, step number 3 in the procedure for keying in the program should read as follows:

"Key in the machine language portion using POKE for the addresses 664DA through 664F9 *and* 66560 through 665CE."

Also, on page 6, keying BRK followed by 4 digits provides a *starting* address argument to DEF H, not an ending address.

In the listing for the Monitor on page 7, change location 64D7 to contain 56 (not 58) and change location 6573 to be 20 (instead of E0).

HP-71B Curve-Fitting Corrections

The ends of several lines in the listing of the Curve-Fitting program in Issue 40 of *PCN* are missing. The last statement in each of the lines listed below should read as indicated:

```
50 ... @ GOTO 20
65 ... ELSE DISP "3-LOG      4-POWER"
105 .. @ IF L<=0 THEN J=1
130 .. @ IF J*T=1 THEN RETURN
390 .. @ C=R*R
```